ChatGPT

BrainDrive – Self-Hosted AI Platform (Full Context)

Introduction and Overview

BrainDrive is an open-source alternative to ChatGPT that you fully **own and control** [1]. It is designed to be **self-hosted** and **MIT-licensed**, giving users complete ownership over their AI – *"Your AI, your rules."* BrainDrive is highly **modular** and customizable: you can extend its functionality through plugins and even monetize your custom AI solutions [1]. The philosophy is similar to *WordPress for AI* – you install the BrainDrive core, then add or develop plugins to quickly ship new AI-powered features [2].

Out of the box, BrainDrive provides a web-based **chat interface** for interacting with AI models, a built-in **Plugin Manager** to easily install or manage extensions, and a **Page Builder** for creating custom user interfaces without coding [3]. It also comes with a suite of developer resources, example plugins, and tutorials to support building on the platform [4]. All of this runs under your control, with **no Big Tech lock-in** – you decide where to host it and how to use it.

Use Cases: BrainDrive can be used as your personal AI assistant and as a foundation to build AI-powered products or services for others. The possibilities are vast – essentially, **"the only limit is your imagination."** For example, you can build [5]:

- **Custom AI chatbots** with specialized knowledge or personalities [6]
- **AI-powered productivity tools** (e.g. assistants that integrate with your data/workflows) [6]
- **Data analysis dashboards** or intelligent reports [7]
- **Custom UIs for specific AI models** (wrapping local or API-based models in user-friendly interfaces) [7]
- **Multi-plugin workflows** where multiple AI components and plugins work together seamlessly [8]

You can run BrainDrive on your local machine or on any cloud server you choose. **Host it locally or anywhere** – there's no dependency on a proprietary service [9]. You can build solutions just for yourself, share with the community, or even offer AI-powered products to customers – **your AI, your rules** [9].

Architecture and Core Components

BrainDrive is composed of a **Core System** and a **Plugin Ecosystem** [10]:

- **Core System (BrainDrive-Core repository):** This includes the **Frontend** (a React+TypeScript web application) and the **Backend** (a Python FastAPI server with a SQLite database by default) [11]. The core provides the primary UI (chat interface, page builder, etc.) and backend services (user management, plugin management APIs, conversation storage, etc.).

- **Plugin Ecosystem:** BrainDrive supports a **plugin architecture** where each plugin is a separate module (hosted in its own repository) that can be dynamically added to the system [12]. The system uses **Webpack Module Federation** to load frontend plugins at runtime, and a standardized

Lifecycle Manager interface (in Python) to integrate backend or installation logic. Plugins can be installed with one click via the Plugin Manager UI or manually, and they communicate with the core through well-defined service interfaces (see **Service Bridges** below) [12]. This decoupled design means you can customize and extend BrainDrive without modifying the core, and update core or plugins independently.

Backend Tech Stack

The BrainDrive **backend** is built with a modern Python web stack [13]:

- **FastAPI** – high-performance Python web framework for the API [14].
- **SQLModel** (on top of SQLAlchemy & Pydantic) – convenient ORM for the SQLite database models [15].
- **Uvicorn** – an ASGI server to run the FastAPI app (used for both development and production deployments) [15].
- **Pydantic** – for data validation and settings management [16].
- **Alembic** – database migration tool to handle schema changes over time [17].
- **SQLite** – the default lightweight database (easy to start with; you can upgrade to a different DB if needed) [18].
- **Structlog** – structured logging for consistent, JSON-formatted logs [19].
- **Passlib** – for secure password hashing (used in authentication) [20].
- **Python-Jose** – for JWT creation and verification (handles auth tokens) [20].

Backend Features & Services: The backend provides robust features to support the AI system [21]:

- **Authentication & Security:** JWT-based auth with access & refresh tokens, role-based access control, and CORS support [21] [22].
- **User Management:** Endpoints for user registration, login, profile management, and a system of “user updaters” that run on each login to handle migrations or updates per user [23].
- **Settings System:** A dynamic settings configuration that supports multi-tier (e.g., system-wide vs user-specific settings) [24].
- **Plugin System:** A modular plugin management system with automatic discovery of plugins. The backend can install/uninstall plugins on the fly and maintains a registry of available plugins [25]. It supports **universal plugin lifecycle APIs** (install, uninstall, status, etc. for any plugin) – see the **Plugin Development** section for details.
- **AI Providers:** Abstraction for AI model providers, making it easy to switch between local models or API-based models. (For instance, integration with an Ollama local model is included as a plugin) [26] [27].
- **Navigation & UI Schema:** The backend helps drive the dynamic UI, providing navigation structure and component manifests to the frontend so that new pages and plugin components can render without hard-coding in the core [28].
- **Conversation Management:** Built-in handling of conversation history, message threads, and related metadata, allowing the chat interface to retain context [29].
- **Tagging & Organization:** A tag-based system to organize conversations or other entities (for example, to label and filter chats or content) [30].
- **Miscellaneous:** The backend also includes features like environment-based configuration (dev vs prod modes), structured logging for debugging, and health endpoints. It is designed with a focus on flexibility, security, and a good developer experience [31] [22].

Frontend Tech Stack

The BrainDrive **frontend** (often referred to as *BrainDrive PluginStudio*) is a single-page web application built with modern web technologies [32] :

- **React 18** – for building a dynamic user interface in a component-driven way [33] .
- **TypeScript** – provides static typing to catch errors early and improve developer productivity [33] .
- **Vite** – a fast development build tool that enables hot-reload and optimized production builds [34] .
- **Material-UI (MUI)** – a UI component framework implementing Google's Material Design, used for a consistent and responsive design out-of-the-box [35] .
- **React Router** – for client-side routing between the different pages (chat, studio, settings, etc.) in the application [36] .
- **React Grid Layout** – for the draggable and resizable grid system in the Page Builder (visual layout editor) [37] .
- **Axios** – for simplifying HTTP requests to the backend API [38] .
- **Zod** – for runtime schema validation of data (especially when dealing with forms and API responses in TypeScript) [39] .

Frontend Features & UI: The BrainDrive UI is designed to be **user-friendly**, **responsive**, and **powerful** [40] :

- **Plugin Management Interface:** A built-in **Plugin Manager** allows users to browse available plugins, install or remove plugins, and manage updates through a visual interface [41] . You can add plugins by providing a GitHub URL and the system handles downloading and installing them.
- **Visual Page Builder:** A **drag-and-drop editor** (PluginStudio) lets you create custom pages by arranging plugin components on a canvas [42] . This visual editor supports multiple layouts (desktop, tablet, mobile) for responsive design [43] . You can configure plugin components via a properties panel and compose complex dashboards or UIs without writing code.
- **Page & Route Management:** Users can create and organize pages, and the system manages navigation routes automatically. You can define multiple pages in your BrainDrive and navigate between them; the routes are handled by the frontend router with persistence in the backend [44] .
- **Component Configuration:** Each plugin's UI components can have configurable properties. The frontend provides forms/UI to adjust these settings per instance of a component (for example, setting a data source or prompt for a chatbot component) [45] .
- **Theme Support:** BrainDrive supports light and dark modes and allows theme customization [45] . The UI will adapt to user preferences, and plugin components can also react to theme changes through the Theme service.
- **User Authentication:** The app includes secure login/logout flows and will hide or show features based on the user's authenticated state and permissions [46] .
- **Real-Time Preview:** When building pages, changes are reflected instantly. The development mode of the frontend supports hot-reloading components so you can see updates without full page refreshes [47] .
- **Import/Export:** Configurations for pages and layouts can be exported to or imported from JSON, making it easy to share setups or back up your designs [48] .
- **Error Handling:** The UI has robust error boundaries to catch and display errors gracefully, and provides feedback if something goes wrong (e.g., plugin fails to load) [48] .
- **Extensible Service Architecture:** The frontend communicates with the backend (and orchestrates plugins) via a set of **service bridges** (see Plugin Development section). This keeps plugin code

decoupled from core code and ensures that even as the system grows, plugins use a stable API surface [49].

The frontend and backend work together: the backend serves the API and plugin lifecycle operations, while the frontend is the interactive layer where users configure and use their AI and plugins. The two communicate primarily over HTTP (REST API), and the system is designed such that you can run both components on the same machine for local use, or deploy them separately if needed.

Installation Guide

BrainDrive is actively tested on **Windows, macOS, and Linux** and the installation process is similar across these platforms [50]. Below are the steps to get BrainDrive running in a **development** setup:

Prerequisites

Before installing, make sure you have the following tools installed on your system [51]:

- **Conda (Anaconda/Miniconda)** – for managing the Python environment [52]. (Alternatively, you can use Python's built-in `venv` or other environment managers, but the guide uses Conda for consistency.)
- **Git** – to clone the BrainDrive repository [53].
- **Node.js (and npm)** – Node.js 16.x or higher is required for the frontend build [54]. (The installer will use npm; yarn is also supported if you prefer.)

  Development Mode Note: When running BrainDrive for development, you will need **two terminal windows** (or tabs): one for the backend server and one for the frontend development server [55].

Step 1: Set Up a Python Environment

It's recommended to use Conda to create an isolated environment for BrainDrive (to avoid version conflicts). For example, to create a Conda environment named `BrainDriveDev` with Python and Node.js, run [56]:

```
conda create -n BrainDriveDev -c conda-forge python=3.11 nodejs git -y
conda activate BrainDriveDev
```

This will create and activate a new environment with Python 3.11, Node.js, and Git installed inside it [57]. (You'll want to activate this environment in both of your terminals whenever working on the project [58].)

Step 2: Clone the Repository

Next, download the BrainDrive-Core code from GitHub. In a terminal, run [59]:

```
git clone https://github.com/BrainDriveAI/BrainDrive-Core.git
cd BrainDrive-Core
```

This will create a directory `BrainDrive-Core` containing the project's source code [60]. All further instructions assume you are in this directory.

Step 3: Install Backend Dependencies

Change into the `backend/` directory and install the required Python packages for the backend [61]:

```
cd backend
pip install -r requirements.txt
```

This uses pip to install all the necessary libraries (FastAPI, SQLModel, etc.) as listed in `requirements.txt` [62].

Backend Configuration: Before running the backend, you need to set up a configuration file. In the `backend/` folder, create a file named `.env` with the necessary environment settings [63]. You can quickly get started by copying the provided template:

```
# From within BrainDrive-Core/backend
cp .env-dev .env       # macOS/Linux
# or for Windows:
copy .env-dev .env
```

This `.env` file contains default development settings (like the host, port, debug flags, etc.) [64]. You can open and adjust the `.env` as needed (for example, to change the server port or other settings), but the defaults should work for a local dev setup. *If you prefer, you can manually create* `.env` *by referring to the* `.env-dev` *file for the necessary variables* [65].

Step 4: Run the Backend Server

With the environment activated and dependencies installed, you can now launch the backend API server. In the first terminal (still in the `backend/` directory), run [66]:

```
uvicorn main:app --host localhost --port 8005
```

This starts the FastAPI server using Uvicorn, listening on `http://localhost:8005` by default [67]. You should see console output indicating the server is running. By default, in development, the server will auto-reload if you edit backend code (because `--reload` is on by default in `.env-dev` settings).

Step 5: Install Frontend Dependencies

Now, open the second terminal window for the frontend. Change into the `frontend/` directory and install the Node.js dependencies (this may take a few minutes) [68]:

```
cd ../frontend   # from the root BrainDrive-Core directory
npm install
```

This will download all the required JavaScript packages for the React app.

Frontend Configuration: Similar to the backend, you should set up a `.env` file in `frontend/`. There is an example file provided. From `BrainDrive-Core/frontend`, run [69]:

```
cp .env.example .env       # macOS/Linux
# or on Windows:
copy .env.example .env
```

This will create a `.env` file for the frontend with default settings. By default, this may include a development convenience like an auto-login token; **make sure to remove any auto-login credentials in** `.env` **before using BrainDrive in production** [70].

Step 6: Run the Frontend (Development Mode)

In the `frontend/` directory (second terminal), start the development server by running [71]:

```
npm run dev
```

This launches the React development server (via Vite). It will open BrainDrive's interface in your browser at `http://localhost:5173` (port 5173 is the default for Vite) [72]. Now the BrainDrive application should be fully running: the frontend will automatically connect to the backend API at localhost:8005.

At this point, you can verify everything is working:

- Open **http://localhost:8005** in a browser – you should see the FastAPI automatic docs (or a JSON message if not using a web browser) confirming the backend is running [73]. You can also access the Swagger UI at **http://localhost:8005/api/v1/docs** for interactive API exploration [74].
- Open **http://localhost:5173** – you should see the BrainDrive web application (login screen or home screen) load in your browser [75]. From here, you can log in (if using default dev config, a test user may be auto-logged in) and start exploring the UI.

If both addresses load as expected, congratulations – you have BrainDrive running locally!

> **Tip:** In the future, after a reboot or stopping the servers, you can restart BrainDrive by re-activating the `BrainDriveDev` Conda environment in two terminals and repeating **Step 4** and **Step 6** (start the backend, then the frontend) [76] [77]. The development servers will pick up where you left off, with all your data (conversations, installed plugins, etc.) preserved in the `backend` folder (e.g., the SQLite database, plugin files, etc.).

Production Deployment (Basic)

The steps above run BrainDrive in development mode (with auto-reload and using the development servers). For a production or persistent deployment, you would:

1. In the `backend/.env` file, set `APP_ENV=prod`, and ensure `DEBUG=false` and `RELOAD=false` to turn off development features [78]. Adjust any other settings (like allowed hosts, etc.) as needed for your environment.
2. Run the backend with a process manager instead of `--reload`. For example, you can use **systemd** or **supervisor** to keep the Uvicorn server running. A suggested Uvicorn command for production is:

   ```
   uvicorn main:app --host 0.0.0.0 --port 8005 --workers 4
   ```

   This binds to all interfaces and uses multiple worker processes for better performance [78].
3. *(If using systemd)* Create a service file to manage the backend. For example:

```
[Unit]
Description=BrainDrive Backend
After=network.target

[Service]
User=BrainDriveAI
WorkingDirectory=/opt/BrainDrive/backend
Environment="PATH=/opt/BrainDrive/backend/venv/bin"
ExecStart=/opt/BrainDrive/backend/venv/bin/uvicorn main:app --host 0.0.0.0 --
port 8005 --workers 4
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

(The above assumes you've installed BrainDrive in `/opt/BrainDrive` and set up a Python virtualenv there.) Enable and start the service with `sudo systemctl enable braindrive && sudo systemctl start braindrive` [79] [80].

1. For the frontend, you can build the optimized production files by running `npm run build` in the `frontend/` directory [81]. This will generate static files in `frontend/dist`. You can then serve these with any static file server or integrate with a web server (like using `vite preview` or copying them to an nginx/apache server). In a simple scenario, you might continue to use the Vite preview or a Node.js static server on port 5173, or incorporate the files into the backend serving. (Detailed production deployment of the frontend will depend on your infrastructure; for a small scale, running `npm run preview` serves the production build locally [82].)

In summary, BrainDrive can be deployed like a typical web application: run the Python backend (perhaps behind a reverse proxy for SSL), and serve the compiled frontend as a static app. In production, you'd typically also set up proper environment variables for any config and ensure the system starts on boot via services.

Using BrainDrive (Basics)

Once installed, BrainDrive is designed to be immediately useful “out of the box.” Upon launching the system, it will open to a working **Chat interface** by default [26]. You can start a conversation with the default AI model. If you have a local model provider configured (for example, the **Ollama** plugin for local large language models), BrainDrive will automatically use it to handle your queries [26] – giving you a private ChatGPT-like experience without requiring external API calls.

From the main interface, you can navigate to the **Plugin Manager** to extend functionality. BrainDrive comes with a few basic plugins (like the chat interface itself is a plugin) and you can install more: - **Installing Plugins:** Go to **Plugin Manager → Install Plugins**, then enter the GitHub repository URL of a BrainDrive plugin to install [83]. (For example, BrainDrive provides a **Plugin Template** repository which you can use as a starting point. By entering its URL and clicking Install, the system will download and add that plugin.) Progress and results of installation are shown in real-time via the interface [84] [85]. Once installed, new plugins become available in the Page Builder and can provide new features or services. - **Enabling/ Disabling:** Through the Plugin Manager, you can enable or disable plugins. Disabling a plugin will remove its UI components from the interface and stop any background tasks it has, without uninstalling it entirely. - **Plugin Updates:** BrainDrive tracks plugin versions. If a plugin repository is updated with a new release, the Plugin Manager will allow you to update to the latest version. This is handled user-by-user, meaning each instance of BrainDrive (or each user on a multi-user setup) controls which plugin versions they are running [86]. - **Plugin Removal:** You can uninstall a plugin via the UI as well, which will cleanly remove its files and data for the current user, thanks to the standardized plugin lifecycle management [87].

The **BrainDrive Studio (Page Builder)** is another core area of the app. Here, you can create custom pages by dragging plugin components onto a canvas. For example, you could create a dashboard page that has a chat component, a data visualization component (from another plugin), and maybe a notes component side by side. You can resize and configure these components visually [40]. Each plugin defines what components it offers and what settings they have (via its manifest), and BrainDrive Studio reads that to present configuration options to you. You can create multiple pages and navigate between them; BrainDrive will remember the pages you create (storing their layout in the backend database) and also allow you to export them as JSON if you want to share your layout with someone else or back it up [47].

Behind the scenes, BrainDrive’s frontend uses a **service-based architecture** to talk to the backend and coordinate plugins. If you’re simply using BrainDrive as an end-user, you won’t need to worry about this. Developers, however, will appreciate that when a plugin component on a page needs something (like to fetch data or save settings), it does so through BrainDrive’s **Service Bridges** rather than directly calling the backend or other plugins. This ensures that plugins remain loosely coupled and that internal APIs can evolve without breaking existing plugins.

Plugin Development Guide

One of BrainDrive’s core strengths is its **extensibility**. Developers can create plugins that add new features, UI components, or AI capabilities to BrainDrive. The system is built to make plugin development as smooth as possible, with a **“1-minute development cycle”** for rapid iteration [88] . Below is a summary of how the plugin system works and how you can start building your own plugin.

Plugin Structure and Lifecycle

A BrainDrive plugin is essentially a self-contained module (usually a Git repository) that includes a frontend part (JavaScript/TypeScript code, typically a bundle that gets federated into the BrainDrive app) and optionally a backend part for any server-side logic. Every plugin has a standard structure and **lifecycle manager**:

- **Standard Structure:** A plugin repository typically has a folder layout like:

```
MyPlugin/
  ├── package.json         # Metadata and dependencies for the plugin
  ├── src/                 # Source code (React components, etc.)
  ├── dist/                # Built files (including a remoteEntry.js for
module federation)
  ├── lifecycle_manager.py # (Optional) Backend lifecycle manager class
  └── README.md            # Documentation for the plugin
```

  The key piece for integration is the `lifecycle_manager.py`. If present, this Python file should define a class that ends with `LifecycleManager` (e.g. `MyPluginLifecycleManager`) which implements standard methods for installation, removal, etc. [89] [90] . This class and its data informs the BrainDrive backend how to install/uninstall the plugin and what components it provides.

- **Universal Lifecycle API:** BrainDrive’s backend provides **universal plugin endpoints** that work for any plugin that follows the lifecycle interface. This means you **do not have to write custom API endpoints** for each new plugin; the core system can handle it. The available REST API endpoints include [91] :

```
POST   /api/plugins/{plugin_slug}/install      # Install a plugin
DELETE /api/plugins/{plugin_slug}/uninstall    # Uninstall a plugin
GET    /api/plugins/{plugin_slug}/status       # Get plugin status
(installed/active)
GET    /api/plugins/{plugin_slug}/info         # Get plugin details/
metadata
POST   /api/plugins/{plugin_slug}/repair       # Repair/reinstall a plugin
GET    /api/plugins/available                  # List all discoverable
plugins
```

  The BrainDrive backend automatically discovers any plugins present in the `backend/plugins/` directory and makes them available via these endpoints (using the plugin’s slug identifier) [92] . This approach provides a **consistent interface** for managing plugins and ensures plugins are **user-scoped** (each user can have their own set of plugins installed without affecting others) [93] . It also means plugin developers get a lot of functionality for free (installation, upgrades, etc. handled by core).

- **Lifecycle Manager Class:** The `lifecycle_manager.py` for a plugin typically defines a class with methods such as `install_plugin(user_id, db)`, `delete_plugin(user_id, db)`, and `get_plugin_status(user_id, db)` [94] [95]. BrainDrive will call these at the appropriate times. For example, when a user triggers an install, the core will import the plugin's lifecycle manager class and run its `install_plugin` method. In these methods, plugin developers can define any setup needed (like initializing database tables or default data for that plugin, or copying files). Many plugins can use a standardized template for these (BrainDrive provides examples in the Plugin Template).

- **Frontend Integration:** On the frontend side, if your plugin is a **frontend plugin** (most are, to provide UI components), it needs to expose a module entry (usually `remoteEntry.js` via webpack federation) and declare its components and metadata in a way the core app can fetch. The specifics are handled by the plugin build process (see the Plugin Template project for details). When installed, the core will include the plugin's bundle and make its React components available in the Page Builder's component library.

BrainDrive's **Module Federation** architecture means that when a plugin is installed, its code is fetched on-demand and integrated into the running app without a full reload. This allows a seamless experience where, for instance, installing a new plugin via the UI will cause its components to immediately appear in the Page Builder component list once installed.

Quick Start: Creating Your Own Plugin

For developers eager to build a new plugin, the recommended path is:

1. **Use the Plugin Template:** BrainDrive provides an official **Plugin Template** repository (often called *PluginTemplate*) that contains the boilerplate code for a basic plugin. Rather than starting from scratch, you can fork or clone this template. To install the template plugin into your BrainDrive for testing, use the Plugin Manager: enter the URL `https://github.com/BrainDriveAI/BrainDrive-PluginTemplate` and click **Install** [83]. This will add a plugin called "Plugin Template" to your system. You can then go to the BrainDrive Studio, create a new page, and drag the "Plugin Template" component onto it to verify that it works [96].

2. **Set Up a Development Workflow:** Clone the Plugin Template repository locally to start developing your plugin. For example:

```
git clone https://github.com/BrainDriveAI/BrainDrive-PluginTemplate.git
MyPlugin
cd MyPlugin
npm install
```

This sets up a local project for your new plugin based on the template [97]. Next, you can configure the build for rapid development. Typically, you will want to output the plugin's build directly into the BrainDrive backend plugins directory to avoid reinstalling on every change. In the `webpack.config.js` of your plugin, there is a commented section for output path. Uncomment and set the output path to point to BrainDrive's `backend/plugins/shared/[YourPluginName]/[version]/dist` directory [98]. This way, when you run `npm run build` in your plugin, it will place the updated code where the BrainDrive backend can serve it immediately, eliminating the need to go through the install process for each change.

3. **Run the Plugin in Dev Mode:** With the above setup, you can make changes in your plugin's code (for example, change some text in the default component) [99], then run a build and simply refresh the BrainDrive page (with cache disabled in your browser during development) to see the changes [100] [101]. This typically allows a **1-minute iteration cycle** where you edit code, build, and see updates, rather than a 10+ minute cycle of re-installing the plugin on every change.

4. **Leverage Service Bridges:** BrainDrive provides six core **Service Bridges** that your plugin can use to interact with the system without needing external dependencies [102]. These are available via the `this.props.services` object in a React component or via imports in backend lifecycle code:

5. **API Bridge:** e.g. `await services.api.get('/some_endpoint')` to call backend APIs (or your own plugin's backend logic) [103].
6. **Event Bridge:** e.g. `services.event.emit('eventName', data)` to send or listen for cross-plugin events (for communication between plugins) [103].
7. **Theme Bridge:** e.g. `services.theme.getCurrentTheme()` to get theme info or subscribe to theme changes (so your plugin can adjust to light/dark mode) [104].
8. **Settings Bridge:** to get or set user preferences/settings, e.g. `services.settings.getSetting('myKey')` [104].
9. **Page Context Bridge:** to get info about the current page or route, e.g. `services.pageContext.getContext()` [105].
10. **Plugin State Bridge:** for persistent plugin-specific storage (key-value store scoped to your plugin), e.g. `services.pluginState.save(data)` [105].

These bridges dramatically simplify development – for example, instead of writing your own API and database calls for your plugin, you might use the *API bridge* to store or fetch data through the BrainDrive backend, or use *Plugin State* to persist small amounts of data without setting up your own database. They also ensure that plugins remain forward-compatible; because plugins use these abstracted services, changes in core internals won't break the plugins as long as the service contracts remain stable.

1. **Build Something Simple First:** When starting out, it's recommended to attempt a small project to get familiar. For instance, you can create a "Hello World" AI plugin that sends a prompt to an AI model and displays the response, using the API bridge to call the backend's chat API [106]. Or a widget that changes appearance based on theme, using the Theme bridge [107]. Keep these initial projects simple and incremental – test the unmodified template first, then make small changes one at a time and observe the results [108]. This helps isolate issues and ensures you understand how each part of the system works.

BrainDrive's community is growing, and you're encouraged to share your plugin ideas or get help on the official forum. The architecture might evolve (the project is in active development), but the goal is to maintain these core patterns so that plugins remain compatible even as BrainDrive reaches future versions.

Current Status and Future Roadmap

BrainDrive is currently in an **open beta (v0.6.x)** stage [27] . Many core features – such as the plugin manager, page builder, and a basic chat interface – are operational and being polished. The focus at this stage is to ensure a smooth developer experience and core functionality for early adopters. For example, recent versions introduced a unified **Dynamic Page Renderer** (for consistently loading plugin UIs) and improvements to user onboarding (registration flow) [109] [110] . Some aspects like responsive design of certain components are still being refined (e.g., making sure pages built with the editor work well on all screen sizes) [110] .

Looking forward, the roadmap to **Version 1.0** includes plans for: - A **One-Click Installer** for non-technical users (to simplify installation on each platform) [111] [112] . - An in-app **Dashboard & Notifications** system to inform owners about system status, available updates, or recent plugin activities (planned for v0.6.5) [113] . - More **polish and case studies** (v0.7–0.9) to ensure the system is robust and demonstrating real-world use cases – for instance, creating example setups that a non-technical user like "Katie Carter" could use out-of-the-box [114] . - A strong **Concierge** or guided experience that can answer common questions and help new users get started with their personal AI (likely by 1.0 launch) [115] . - Cross-platform support and testing to guarantee BrainDrive runs on Windows, Mac, and Linux smoothly without requiring developer assistance [116] .

The ultimate **mission** of BrainDrive is *"to make it easy to build, control, and benefit from your own AI system."* [117] All development is guided by this goal, focusing on empowering end-users and developers alike to create useful AI applications under their own ownership.

Community and Support

BrainDrive is not just a product, but also a community project. Here are resources for help and discussion:

- **Community Forum:** The primary place for asking questions, getting help, and discussing ideas is the BrainDrive forum (community.braindrive.ai) [118] . Both end-users and developers hang out there. The development team posts weekly progress updates and encourages feedback in specific threads [119] . If you run into issues installing or using BrainDrive, chances are someone on the forum can assist.
- **GitHub Issues:** For bug reports or feature suggestions, you can open an issue on the GitHub repository. If your issue is related to a plugin, please tag it with the plugin name (e.g., `[plugin]`) in the title for clarity [120] . The maintainers monitor these and will respond as time permits.
- **Documentation Index:** The BrainDrive GitHub organization contains multiple repositories (Core, Plugin Template, example plugins, etc.). There is a **Documentation Index** linking to all available docs and guides [121] . This is a great starting point if you're looking for more in-depth tutorials or reference material on specific aspects (like customizing the Lifecycle Manager, or examples of each Service Bridge in action).
- **Direct Contact (Security):** If you believe you've found a security vulnerability in BrainDrive, please do **not** post it publicly. Instead, use GitHub's private vulnerability reporting or email the core team

directly as outlined in the Security Policy [122] [123]. They commit to prompt responses for security issues.

BrainDrive's tagline is **"Your AI. Your Rules."** – the community truly believes in user ownership. Feedback and contributions are welcome to improve the project for everyone.

Contributing and Project Status

BrainDrive is in an early phase (developer beta), which means the core team is actively making changes and some parts of the system may not be fully stable yet for general use [124]. At this stage, the best way to contribute is to try out the system, **report your experience**, and maybe help with small fixes or improvements. The project welcomes early technical adopters who share the vision of personal AI to provide feedback and help shape the roadmap [125] [126].

If you're interested in contributing code or plugins: - Keep an eye on the repository for updates. The maintainers will be sharing contribution guidelines and more developer tools as the project matures [127]. Soon, they plan to open up for broader contributions to both the core and the plugin ecosystem. - Be aware that **things are moving fast** – internal APIs or architectures might change while in beta, so expect that some plugins or integrations you write now might need adjustments as we approach v1.0 [128]. Major components like the renderer and plugin system are still being refined for stability and performance. - Follow the four **Guiding Principles** that steer the project [129]: 1. **Ownership** – Users should ultimately own their BrainDrive data and setup. 2. **Freedom** – Avoid lock-in or restrictive designs; the project won't introduce proprietary traps. 3. **Empowerment** – Documentation and design should empower others to use and build on BrainDrive easily. 4. **Sustainability** – Focus on contributions that help the ecosystem thrive in the long run. - **Long-Term Vision:** The project is currently stewarded by BrainDrive LLC, but the goal is to gradually decentralize development and governance to the community of users, builders, and entrepreneurs as the ecosystem grows [130]. Early contributors are helping to lay the foundation of what could be a large, community-driven platform.

If you have questions about contributing or where to start, you can reach out on the forum or open a GitHub discussion/issue [131]. The maintainers are happy to guide new contributors. And even if you're not coding, just being an early user and sharing feedback is extremely valuable at this stage.

Troubleshooting

Here are some common issues you might encounter when installing or running BrainDrive, and how to resolve them:

Issue	Solution
Package install fails	Run `pip install --upgrade pip` and retry the install [132].
Port 8005 or 5173 in use	Another process is using the port – edit the `.env` to change the `PORT` (for backend) or adjust the frontend dev server port, then restart [132].

Issue	Solution
Module not found error	A required Python package might be missing – run `pip install <module>` (or add it to `requirements.txt`) and try again [133].
Database errors	Check that the `.env` DB settings (like file paths) are correct. The default uses SQLite – ensure the app has write access to the folder [134].
Environment activation issues	If `conda activate BrainDriveDev` fails, make sure Conda is installed and the environment was created. For virtualenv, ensure you've activated the correct venv [135].
Frontend not updating	In dev, ensure your browser cache is disabled (open dev tools and check "Disable cache") to see live updates. Also verify both servers are running with no errors in their consoles.
Plugin installation fails	Double-check the GitHub URL and that the repository has a release/build. Look at the backend console for error logs during plugin install – it often logs what went wrong (e.g., network issue, or plugin structure invalid).

(For more detailed support, refer to the community forum or GitHub issues as mentioned above.)

License

BrainDrive is released under the **MIT License** [136]. This permissive open-source license means you are free to use, modify, and distribute BrainDrive as you see fit, with no warranty implied. Essentially, *you own your BrainDrive* – both in terms of data and the software itself.

Your AI. Your Rules. [137]

---

[1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [88] [118] [120] [121] [136] README.md
https://github.com/BrainDriveAI/BrainDrive-Core/blob/e5a26ec775d016d55d350c41d464807ecb127d33/README.md

[13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23] [24] [25] [28] [29] [30] [31] [74] [78] [79] [80] [132] [133] [134] [135] README.md
https://github.com/BrainDriveAI/BrainDrive-Core/blob/e5a26ec775d016d55d350c41d464807ecb127d33/backend/README.md

[26] [27] [109] [110] [111] [112] [113] [114] [115] [116] [117] [119] ROADMAP.md
https://github.com/BrainDriveAI/BrainDrive-Core/blob/e5a26ec775d016d55d350c41d464807ecb127d33/ROADMAP.md

[32] [33] [34] [35] [36] [37] [38] [39] [40] [41] [42] [43] [44] [45] [46] [47] [48] [49] [81] [82] README.md
https://github.com/BrainDriveAI/BrainDrive-Core/blob/e5a26ec775d016d55d350c41d464807ecb127d33/frontend/README.md

[50] [51] [52] [53] [54] [55] [56] [57] [58] [59] [60] [61] [62] [63] [64] [65] [66] [67] [68] [69] [70] [71] [72] [73] [75] [76] [77] INSTALL.md
https://github.com/BrainDriveAI/BrainDrive-Core/blob/e5a26ec775d016d55d350c41d464807ecb127d33/INSTALL.md

[83] [96] [97] [98] [99] [100] [101] [102] [103] [104] [105] [106] [107] [108] PLUGIN_DEVELOPER_QUICKSTART.md
https://github.com/BrainDriveAI/BrainDrive-Core/blob/e5a26ec775d016d55d350c41d464807ecb127d33/PLUGIN_DEVELOPER_QUICKSTART.md

[84] [85] README.md
https://github.com/BrainDriveAI/BrainDrive-Core/blob/e5a26ec775d016d55d350c41d464807ecb127d33/frontend/src/features/plugin-installer/README.md

[86] [87] [89] [90] [91] [92] [93] [94] [95] UNIVERSAL_LIFECYCLE_GUIDE.md
https://github.com/BrainDriveAI/BrainDrive-Core/blob/e5a26ec775d016d55d350c41d464807ecb127d33/backend/app/plugins/UNIVERSAL_LIFECYCLE_GUIDE.md

[122] [123] SECURITY.md
https://github.com/BrainDriveAI/BrainDrive-Core/blob/e5a26ec775d016d55d350c41d464807ecb127d33/SECURITY.md

[124] [125] [126] [127] [128] [129] [130] [131] [137] CONTRIBUTING.md
https://github.com/BrainDriveAI/BrainDrive-Core/blob/e5a26ec775d016d55d350c41d464807ecb127d33/CONTRIBUTING.md